Pixel Processor

The pixel processor receives raw data from the pixel chips and formats it for use by the hit processors and level-2 trigger. It processes hits for an entire quadrant of a pixel plane, and is capable of handling 30 million hits per second. The following tasks are done by the pixel processor:

1) Parse the pixel data stream to construct hit records and combine adjacent hits into a single hit cluster (in the narrow pixel dimension).
2) Merge multiple hit streams into a single stream.
3) Translate a hit's (Chip, Row, Col) to system coordinates (X, Y, Z) and assign it to a processing zone (Phi slice).
4) Compute the center of mass of the hit cluster and add it to the coordinates.
5) Store hits into delay buffer by timestamp. Send events from delay buffer to hit processor. Hold events in delay buffer for level-2 readout.

Parsing

The protocol for transmitting pixel data is not yet completely defined, however it will at least contain the fields shown in figure 2. It may have a fixed format where all fields are sent for every hit, or it may have a variable format where a context (e.g. Time and Dev) applies to a variable number of hits. The parser will extract this information and use it to create Fixed-format hit records.

The parser will also combine several adjacent hits into clusters. Again, the format has not been completely specified. It's possible that the pixel chips will send only a single pulse height per hit. It's also possible that it will send all pulses that belong to a 4-pixel group. A simple state machine can handle both cases and assign hits to the proper cluster. This compares the incoming hit to the current (T, D , C, and R+1). If they are the same, the incoming hit is combined with the current hit.

Merging

It's possible that the data for a quadrant will be carried by more than 1 data link. In this case, a Parser unit is required for each link, and all parser outputs must be merged into a single data stream.

Translate

The Dev, Col, and Row fields are used as an index into a lookup table which provides X, Y, Z, and P. X and Y are 15 bits each and represent the hit's location to 1/32nd the width of a pixel. Z is a single bit that indicates what side of the plane the pixel chip is on. P is a 4-bit number that indicates what processing zone(s) the hit belongs to. In the baseline design, hits are assigned to one of 8 pie-shaped processing zones with the 4th bit used to indicate overlap.

A second table keeps track of pixel hits. A bit is set in this table for each pixel that is hit. This will be used to monitor dead pixels.

Fig 1. Pixel Processor

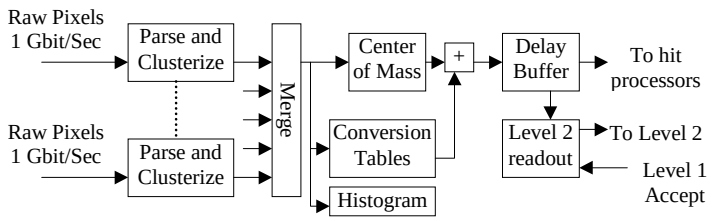
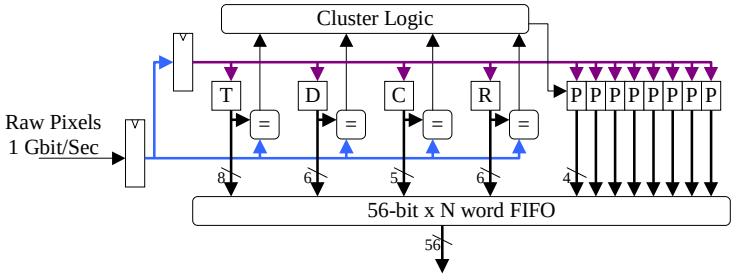


Fig 2. Data found in raw pixel stream

Name	Size	
Time	6-8	Time stamp.
Dev	6	Device (Chip) number.
Col	5	Column number.
Row	8(6)	Row number or (cluster number)
Pulse	4(16)	Pulse heights. Assuming a 4-bit ADC. Possibly 4 pulses per cluster.

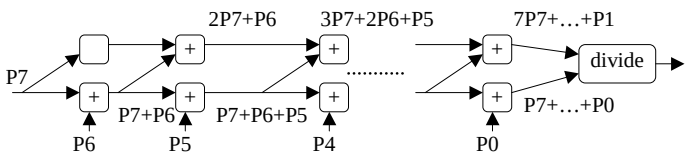
Parse and Clusterize



Center of Mass

The center of mass computed over 8 pulse heights will be a number from 0.0 to 7.0 that indicates the center of a cluster relative to the center of pixel 0. In an FPGA, the calculation can be done with a pipelined adder chain and produce a new result avery 20 ns. The divider is implemented with small lookup tables and can produce a result accurate to 6 or 7 bits.

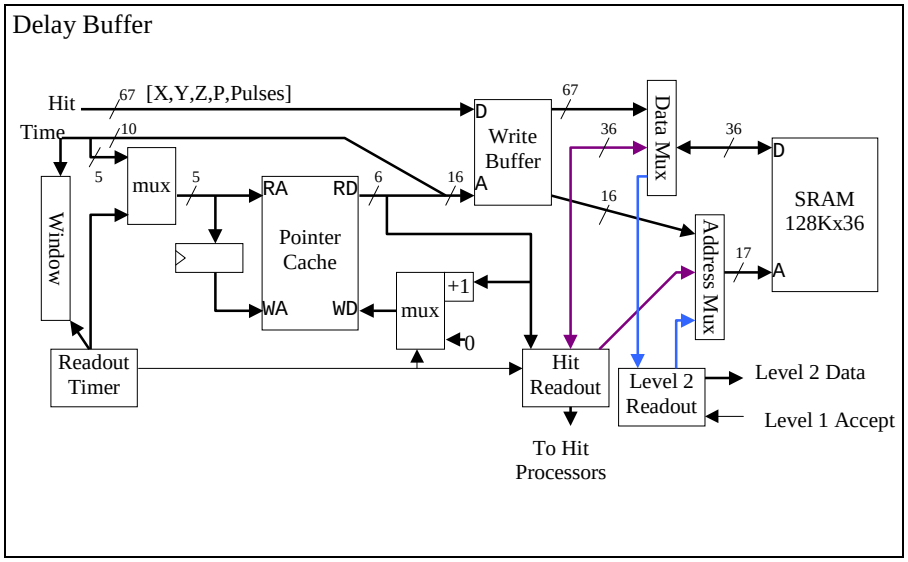
$$C.O.M = \frac{\sum_{n=0-7} n * Pn}{\sum_{n=0-7} Pn}$$



Delay Buffer

The delay buffer can hold 64 hits for each of 1024 bunch crossings. A 36-bit wide SRAM is used to provide a bandwidth of 100 Megawords/Sec. 60% of this bandwidth is used for writing hits (67 bits @ 30 MHz). Another 30% is used for transferring hits to the hit processors (35 bits @ 30 MHz. The raw pulse height data is not sent to the hit processors.) The remaining bandwidth is available for level-2 readout and pointer updates. (Note that the delay buffer is the bottleneck of the design. The rest of the pixel processor can handle hits at a 50MHz sustained rate. If necessary, the speed of the delay buffer can be doubled by adding a second SRAM bank. However, this would more than double its complexity)

As shown in the diagram, the lower 5 bits of a hit's timestamp are used as an index to a pointer cache. The associated 6-bit pointer contains a count of the number of hits that have arrived for that time stamp. The count is combined with the 10-bit time stamp to generate a 16-bit address for writing the hit to memory. The 6-bit pointer is incremented and written back to the pointer cache.



A pointer-cache entry is valid for 20 bunch crossings (2.6 uS). When it becomes older than 20 bunch crossings, it is sent to the hit readout circuit and a 0 is written back to the cache. The hit readout uses the pointer to write an END flag to the SRAM, and then copy the event to the hit processors. Note that if the hit count is 0, the END flag will still be written, but no data will be sent to the processors. The END flag is used by the level-2 readout.

A window circuit will reject any hits that are older then 20 bunch crossings. A count of rejected hits will be kept for diagnostic purposes. Note that the actual delay can be set to any value between 1 and 30 and should be tuned to the smallest number that will accept 99% of the hits.

Level-1 accept signals are received via a serial link and contain a 10-bit timestamp. When an accept is received, the timestamp is multiplied by 64 and used as an index into the delay buffer to locate the start of the event's data. Hits are read sequentially from this location until the END flag is found.